ANNO 3 - 13 - L. 5000 - Frs. 7.50

PC 1 - PC 128S - PC 128

olivetti PRODEST USER

LA PRIMA E UNICA RIVISTA INDIPENDENTE PER GLI UTENTI PC 128-PC 128S-PC 1

Trucchetti con il GWBasic

Wordstar

PC 1

Comunicare con la RS232

PC 128S

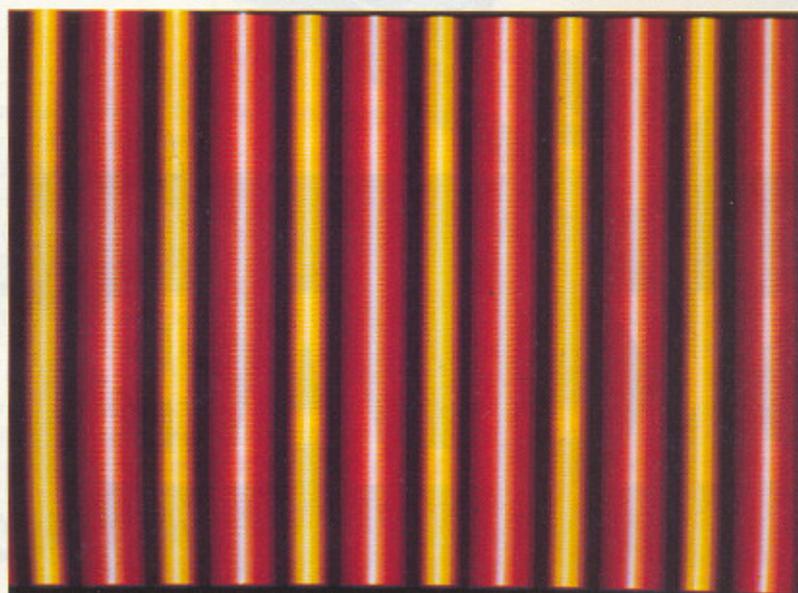
Astronomia

PC 128

Costruzione di grafici

MODEM

Spedizione in abbonamento postale Gruppo III/70 Tassa pagata (Tassa riscossa) Milano Ferr. Corr.

GRUPPO EDITORIALE JACKSON

I DUE BASIC del PC 128

ULTIMA PUNTATA

Con questa puntata si conclude la serie di articoli dedicati ai due Basic del PC 128, articoli che ci hanno seguiti fin dalla nostra prima uscita in edicola.

Nella speranza che il nostro sforzo vi sia stato di una qualche utilità, vi diamo ora appuntamento per scoprire assieme altre meraviglie del PC 128.

SQR

TIPO: Funzione matematica

SINTASSI: SQR(n.)

COMMENTO:
Dove n. e' un numero o una variabile numerica, positivo.

La funzione SPQ(), ritorna la radice quadrata dell'argomento.

Se n. e' un numero negativo, il computer segnala un errore di tipo "Illegal Function Call".

```
10 CLS
20 INPUT"Immettere un numero ",A
30 IF A<0 THEN PRINT"E' un numero negativo,
   ripeti!":GOTO 20
40 CLS
50 PRINT"La radice quadrata di";A;
   "e': ":PRINT
60 PRINT SQR(A)
70 END
```

STEP

TIPO: Funzione

SINTASSI: STEP(n.)

COMMENTO:
La funzione STEP, se usata con i comandi grafici (BOX, BOXF, CIRCLE, CIRCLEF, LINE, PAINT PSET), permette di attuare uno spostamento relativo, calcolato sull'ultimo punto definito. Gli spostamenti possono essere sia negativi che positivi.

```
10 CLS
20 BOXF(100,100)-STEP(-50,50)
```

```
BOXF(200,150)-STEP(40,40)
30 END
```

STICK

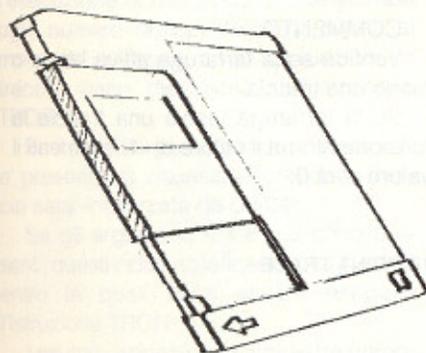
TIPO: Funzione

SINTASSI: STICK(n.)

COMMENTO:
Dove n. e' un numero che puo' valere 0 o 1.
La funzione STICK() ritorna la direzione del joystick specificato in argomento.
I valori ritornati dalla funzione sono:

0 neutro
1 90o
2 45o
3 0o
4 -45o
5 -90o
6 -135o
7 180o
8 135o

Vediamo un piccolo programma dimostrativo:

```
10 CLS
```

```
20 X=19
30 Y=12
40 A=STICK(0)
50 IF STRIG(0)=-1 THEN END
60 LOCATE X,Y
70 PRINT" "
80 ON A GOSUB 160,180,210,230,260,280,
   310,330
90 IF X<0 THEN X=0
100 IF X.39 THEN X=39
110 IF Y<0 THEN YY=0
120 IF Y>24 THEN Y=24
130 LOCATE X,Y
140 PRINT"X";
150 GOTO 40
160 Y=Y-1
170 RETURN
180 Y=Y-1
190 X=X+1
200 RETURN
210 X=X+1
220 RETURN
230 Y=Y+1
240 X=X+1
250 RETURN
260 Y=Y+1
270 RETURN
280 Y=Y+1
290 X=X-1
300 RETURN
310 X=X-1
320 RETURN
330 Y=Y-1
340 X=X-1
350 RETURN
```

STOP

TIPO: Istruzione

SINTASSI: STOP

COMMENTO:
Sospende temporaneamente l'esecuzione del programma, e fa apparire sullo schermo il messaggio "Break in ...".
E' possibli riprendere l'esecuzione del programma, per mezzo dell'istruzione CONT, oppure con GOTO.

STR$

TIPO: Funzione

SINTASSI: STR$(n.)

COMMENTO:
Dove n. e' un numero o una variabile numerica consentita dal Basic.
L'istruzione STR$, converte il numero in argomento in una stringa.
La funzione inversa di STR$ e' VAL.

```
10 CLS
20 A$=STR$(1234567890)
30 PRINT A$
40 END
```

STRIG

TIPO: Funzione

SINTASSI: STRIG(n.)

COMMENTO:
Dove n. e' un numero o una variabile, il cui valore e' 0 oppure 1.
Ritorna lo stato del pulsante del joystick in argomento.
I valori di ritorno della funzione STRIG() sono:

-1 pulsante premuto
0 pulsante non premuto.

```
IF STRIG(O)=-1 THEN END
```

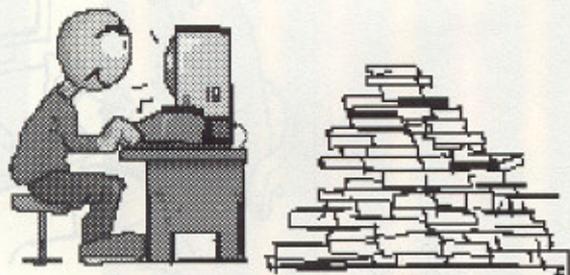
STRING$

TIPO: Funzione

SINTASSI: STRING$(n.,car.)

COMMENTO:
Dove n. e' un numero o una variabile permessa dal Basic; car. e' un carattere o il numero ASCII corrispondente.

La funzione fornisce una stringa di lunghezza n. i cui caratteri sono car., o hanno il codice ASCII di car.

```
10 CLS
20 FOR A=1 TO 10
30 FOR B=65 TO 75
40 PRINT STRING$(A,B)
```

```
50 NEXY B
60 NEXT A
70 END
```

SWAP

TIPO: Istruzione

SINTASSI: SWAPvar1,var2

COMMENTO:
Dove var1 e var2 sono due variabili o elementi di matrici.
Scambia il valore di due variabili dello stesso tipo.
E' possibile scambiare qualsiasi tipo di variabile (intero, precisione singola o doppia, stringa), ma le due variabili devono essere dello stesso tipo.

```
10 CLS
20 A$="10": B$="20": C$="X"
30 PRINT A$;C$;B$
40 SWAP A$,B$
50 PRINT A$;C$;B$
60 END
```

TAB

TIPO: Funzione

SINTASSI: TAB(n.)

COMMENTO:
Dove n. e' un numero o una variabile, consentito dal Basic.
La funzione TAB() permette di posizionare il cursore nella colonna indicata dall'argomento. Se tale colonna si trova a sinistra dell'attuale posizione del cursore, si avra' un ritorno di carrello. Se n. e' un valore nullo o negativo, la funzione non dara' alcun effetto.
Se si sta lavorando con caratteri in doppia larghezza, la funzione TAB() ne tiene conto.
I separatori, virgola e punto e virgola, vengono usati come per l'istruzione PRINT normale.

```
10 CLS
20 PRINT "Ciao"; TAB(15);"Pippo"
30 PRINT "Ciao", TAB(15)'"Pippo"
40 END
```

TAN

TIPO: Funzione matematica

SINTASSI: TAN(n.)

COMMENTO:
Dove n. e' un angolo espresso in radianti. Per convertire i gradi in radianti, moltiplicare n. per P/180, dove P=3,141593.
Calcola la tangente trigonometrica di n.

```
10 CLS
20 SCREEN1,6,6
30 LINE(0,100)-(319,100),2
40 LINE(0,50)-(319,50),4
50 LINE(0,150)-(319,150),4
60 FOR I=0 TO 319
70 AN=I/30
80 Y=100-50*TAN(AN)
90 IF Y<0 OR Y>199 THEN 110
100 PSET(I,Y),1
110 NEXT I
120 END
```

TRACE

TIPO: Funzione

SINTASSI: TRACE

COMMENTO:
Verifica se la tartaruga attiva lascia o meno una traccia.
Se la tartaruga lascia una traccia la funzione ritorna il valore di -1, altrimenti il valore e' di 0.

```
10 PRINT TRACE
```

TRACE

TIPO: Istruzione

SINTASSI: TRACEn.

COMMENTO:
Dove n. e' un numero che vale 0 oppure 1.
Permette di attivare (n.=1), oppure

disattivare (n.=0), la traccia lasciata dalla tartaruga attiva durante i suoi spostamenti.

```
TRACE 1
```

TROFF

TIPO: Istruzione

SINTASSI: TROFF

COMMENTO:
Annulla l'effetto dell'istruzione TRON, e pertanto chiude la traccia e, se aperto, il file di traccia.

TRON

TIPO: Istruzione

SINTASSI: TRONdescrittore file,n.1-n.2

COMMENTO:
Esegue la traccia dell'esecuzione delle specifiche di programma.

Utile come aiuto per la rivelazione degli errori, il comando TRON consente l'esecuzione di una traccia che stampa ogni numero di riga del programma al momento dell'esecuzione. La funzione di traccia viene disattivata dall'istruzione TROFF.

Se il primo argomento, descrittore file, e' presente, la visualizzazione della traccia sara' indirizzata da questo.

Se gli argomenti n.1 e n.2 sono presenti, questi indicano le linee di programma entro le quali deve essere eseguita l'istruzione TRON.

I risultati, appaiono racchiusi tra parentesi quadre.

```
10 CLS
110 TRON
120 IF A>0 THEN IF A>100 THEN 150 ELSE
160 ELSE 170
150 PRINT "A100":END
160 PRINT "0<A<100":END
170 PRINT "A<=0"
180 TROFF
190 END
```

TUNE

TIPO: Istruzione

SINTASSI: TUNE

COMMENTO:
Permette la regolazione della penna ottica; nel Basic 1.

Una volta attivata l'istruzione, ponendo la penna ottica a contatto dello schermo, su questo apparira' una linea verticale: se questa non coincide esattamente con la posizione della penna, la si deve spostare per mezzo dei tasti cursore.

TURTLE

TIPO: Istruzione

SINTASSI: TURTLEn.,cl.,ri.,str.

COMMENTO:
Dove n. e' un numero o una variabile numerica, ed indica il numero della tartaruga attiva; cl. e ri., rispettivamente colonna e riga, determinano le coordinate della posizione della tartaruga sullo schermo, e sono compresi tra -32768 e 32768. Questi due parametri non sono obbligatori, e nel caso vengano omessi, la posizione della tartaruga non viene cambiata, oppure, se la tartaruga e' appena stata attivata, questa sara' visualizzata nel centro dello schermo.

Il quarto parametro, str., e' una stringa, o una variabile stringa, di caratteri che permette di definire la forma della tartaruga. All'interno, la stringa contiene delle coppie di valori che hanno la funzione di tracciare la forma della tartaruga stessa.

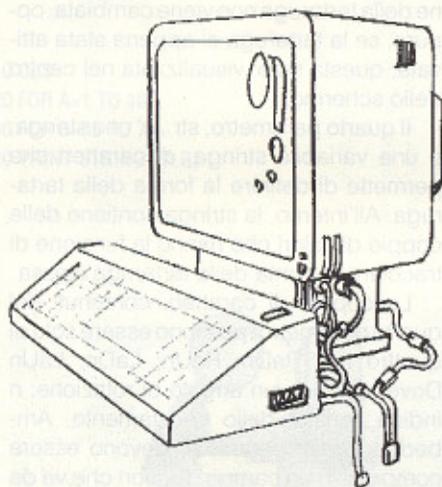
Le coppie di caratteri, contenuti dal quarto parametro, possono essere solo di quattro tipi: RaDn; RaUn; LaDn; LaUn
Dove: a indica un angolo di rotazione; n indica l'entita' dello spostamento. Ambedue questi parametri, devono essere compresi in un campo di valori che va da 0 a 255. Per il parametro a, il valore di 256 corrisponde a 360 gradi.

R Indica rotazione a destra.
L Indica rotazione a sinistra.
D Indica spostamento con traccia.
U Indica spostamento senza traccia.

L'istruzione TURTLE, permette di attivare e posizionare una tartaruga attiva.

```
10 CLS
20 TURTLE1,180,120,"L0D30L90D30L83D26"
30 SHOW1
40 END
```

Visualizza una tartaruga a forma trian-

golare, posta nelle coordinate 180, 120.

UNLOAD

TIPO: Istruzione

SINTASSI: UNLOADn.drive

COMMENTO:
Chiude tutti i file aperti sul dischetto contenuto dal drive specificato in n.drive, e ricopia tutte le informazioni utili sul dischetto.

Il drive di default e' il drive 0, oppure l'ultimo defi- nito per mezzo dell'istruzione DEVICE.

UNLOAD1 chiude tutti i file aperti sul dischetto inserito nel drive 1.

USR

TIPO: Funzione

SINTASSI: USRn.(dati)

COMMENTO:
Dove n. e' un numero compreso tra 0 e 9 e corrisponde alla cifra fornita con la specifica DEFUSR per il programma desiderato. Se si omette n., si assume USR0.

La funzione USR viene usata per richiamare il sottoprogramma in linguaggio macchina definito dall'utente.

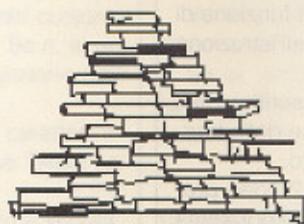
Si fa presente che l'uso approfondito di questa, come di altre funzioni, non puo' far parte di questa trattazione, pertanto se ne consiglia l'uso solo ai programmatori esperti.

VAL

TIPO: Funzione

SINTASSI: VAL(str.)

COMMENTO:
Ritorna il valore dei primi numeri della stringa str. Questo a condizione che nessun segno (esclusi il - e il +), o carattere, preceda il numero.

```
10 CLS
20 PRINT VAL("0234X4")
30 PRINT VAL("-7,4ZE")
40 PRINT VAL("A123")
50 PRINT VAL("*901")
60 PRINT VAL("-3.45P3")
70 PRINT VAL("+3.45P3")
80 END
```

VARPTR

TIPO: Funzione

SINTASSI: VARPTR(var.)

COMMENTO:
Dove var e' il nome di una variabile: numerica; stringa o dell'elemento di schiera del programma.

Il risultato della funzione e' un numero compreso tra -32768 e 32768. Quando il risultato e' un numero negativo, e' necessario aggiungere 65536 per ottenere l'indirizzo reale.

I valori sopra descritti, si riferiscono a un indirizzo, della memoria Basic, relativo al primo byte del dato identificato con var.

Solitamente VARPTR e' usato per ottenere l'indirizzo di una variabile o di una array da passare a sotto programmi in linguaggio macchina.

```
10 CLS
20 A=12345:A$="CIAO PIPPO":B=4567
30 PRINT VARPTR(A)
40 PRINT VARPTR(A$)
50 PRINT VARPTR(B)
60 END
```

VERIFY

TIPO: Istruzioni

SINTASSI: VERIFY ON
VERIFY OFF

COMMENTO:

Per mezzo del comando VERIFY ON il computer controlla tutti i dati contenuti su un dischetto. Tramite VERIFY OFF, invece l'opzione viene disattivata.

WAIT

TIPO: Istruzione

SINTASSI:WAITindirizzo, maschera1, maschera2

COMMENTO:

Permette di sospendere l'esecuzione di un programma fino a quando non appare una precisa configurazione di bit all'indirizzo in argomento.

La lettura dei dati e il modo di usarli e' il seguente:

-Viene fatto un'operazione XOR tra il contenuto dell'indirizzo e la maschera 2.

-Viene fatta un'operazione AND tra il contenuto dell'indirizzo e la maschera1.

Il programma prosegue se queste espressioni danno un risultato nullo.

Dopo questa semplice e semplicistica spiegazione, e' opportuno specificare che anche questa istruzione fa parte di quel campo di comandi utilizzabile solo dagli esperti, e pertanto esula un po' dagli intenti della nostra trattazio- ne.

WINDOW

TIPO: Istruzione

SINTASSI:WINDOWA(cl.1,ri.1)-(cl.2, ri2)

COMMENTO:

Dove cl.1 e ri.1, sono dei numeri o delle variabili numeriche consentite dal Basic, che rappresentano le coordinate di un punto, corrispondente al vertice superiore sinistro di un rettangolo. L'angolo inferiore destro, invece, viene identificato dalla copia di valori contenuta in cl.2 e ri.2.

Il valore di cl. e' definito in un campo che va da 0 a 319 (oppure 639), mentre il valore di ri. e' compreso tra 0 e 199.

Per mezzo dell'istruzione WINDOW, si puo' definire una finestra di visualizzazione grafica.

Una volta specificata l'ampiezza della finestra, le istruzioni di tracciamento grafico, saranno visualizzabili solo al suo interno.

```
10 CLS
20 WINDOW(100,100)-(200,180)
30 CIRCLEF(130,130),49,2
40 END
```

WRITE#

TIPO: Istruzione

SINTASSI: WRITE#n.canale,elenco dati

COMMENTO:

Dove n.canale e' il numero con il quale e' stato aperto il file per l'emissione;

elenco dati, e' una lista di espressioni stringa e/o numeriche separate da virgole.

L'istruzione WRITE# permette di scrivere i dati in un file sequenziale.

Nel caso di file ad accesso diretto, WRITE# mette i dati nel buffer di registrazione. Il trasferimento poi, viene fatto per mezzo della specifica PUT.

La differenza tra WRITE# e PRINT#, e' che WRITE# inserisce le virgole tra gli elementi quando vengono scritti e delimita le stringhe con apici.

```
WRITE#1,Prova1,PROVE
```

ZOOM

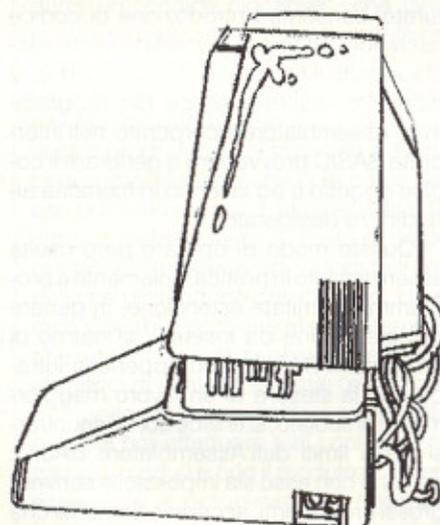
TIPO: Funzione

SINTASSI: ZOOM

COMMENTO:

Ritorna la dimensione della tartaruga attiva. Tale risultato e' un numero compreso tra o e 255.

Per default, la dimensione di una tartaruga definita per mezzo della specifica TURTLE, e' 16.

```
PRINT ZOOM
```

ZOOM

TIPO: Istruzione

SINTASSI: ZOOM TO n.

COMMENTO:

Fissa o modifica le dimensioni della tartaruga attiva.

Se TO e' presente la dimensione della tartaruga e' determinata in modo assoluto, se invece TO e' omesso, la dimensione della tartaruga viene incrementata del valore in argomento.

Il valore di n. e quindi la dimensione, puo' variare da 0 a 255. Seguendo il valore del secondo parametro di SHOW, la dimensione della tartaruga viene modificata immediatamente (se questo valore e' uguale a 1, ossia corrispondente a tartaruga libera), oppure solo dopo FWD o TURTLE.

TURTLE1: ZOOM TO 25 Fissa la dimensione della tartaruga 1 a 25.

ZOOM 30 Aumenta di 30 la dimensione della tartaruga attiva.

UN PACKAGE per il L.M. del PC 128S

A.D.E. The ultimate assembly language development tool System Software ltd.

Programmare in linguaggio macchina con il PC128S è almeno in linea di principio, abbastanza facile (ammesso di conoscere il microprocessore 6502 ed i punti d'ingresso del sistema operativo): il BASIC BBC, già di per sè abbastanza strutturato, consente l'introduzione di codice assembly "in linea", semplicemente racchiudendo il tutto tra parentesi quadre; al momento dell'esecuzione del programma, l'Assemblatore incorporato nell'interprete BASIC provvederà a generare il codice oggetto e ad inserirlo in memoria all'indirizzo desiderato.

Questo modo di operare però risulta essere limitato in pratica, solamente a programmi di limitata estensione, in genere piccole routine da inserirsi all'interno di programmi BASIC. Non appena si intraprende la stesura di un lavoro maggiormente articolato, si finisce con lo scontrarsi con i limiti dell'Assemblatore BASIC; non che con esso sia impossibile scrivere grossi programmi, accade solamente che si comincia a sentire la mancanza di quelle facilitazioni presenti negli assemblatori più "seri" e che consentono di sbrogliare matasse aggrovigliate (o meglio, consentono, a chi ne fa buon uso, di evitare i grovigli!).

A questo punto entra in ballo ADE plus: si tratta di un pacchetto integrato per lo sviluppo di programmi in linguaggio assembly, comprendente, udite udite, un Text Editor, un Macro Assemblatore, un Linker, un Debugger, e per finire ma non certo ultima, una Memory Management Unit (MMU), che funge da interfaccia utente, ed in un certo senso, fa da "colla" per legare assieme tutte queste cose. Non tutti però sapranno esattamente che cosa siano un Macro Assemblatore od un Linker, vediamo dunque un po' più in dettaglio di che si tratta.

Gli strumenti

La funzione principale di un Assemblatore, è quella di tradurre i codici mnemonici che compongono un programma scritto in linguaggio assembly nei valori numerici corrispodenti, sì da produrre un "codice oggetto", direttamente eseguibile dal microprocessore (a questo proposito, ADE+ supporta i mnemonici e la sintassi standard della serie 65C00). Il "codice sorgente", ovvero il testo del programma, viene scritto per mezzo di un text editor (ADE+ ne ha uno incorporato, ma si può usare anche VIEW), dunque salvato su di un file, oppure in un'apposita area di memoria (l'input buffer); a questo punto bisogna invocare l' Assembler, che compirà puntualmente il suo lavoro, segnalando eventuali errori incontrati: ADE+ dispone di una settantina di messaggi d'errore, che si dividono in fatali, non fatali e "warnings", cioè errori che non pregiudicano l'assemblazione, ma meritano comunque di essere investigati.

Al momento di lanciare l'Assemblatore, è possibile utilizzare delle "opzioni", per condizionare lo svolgimento dell'assemblazione (se effettuare o meno il listato del programma, se dirigere l'output sulla stampante, se utilizzare il set d'istruzioni ridotto....); è inoltre possibile specificare il nome del file dove si desidera che vada il codice oggetto prodotto, o quello dell'"output file", dove verranno registrati tutti i risultati del processo d'assemblazione, compreso l'elenco dei simboli incontrati ed un sommario degli errori.

Già questo è molto più di quanto offra l'Assemblatore del BASIC: si ha l'impressione di essere trattati in guanti bianchi: ADE+ ci informa di tutto quello che è successo in macchina e del perché sia successo; ma il bello deve ancora venire. Un asssemblatore che si rispetti infatti, non deve solo riconoscere i codici mnemonici che corrispondono ad istruzioni del microprocessore, ma anche delle "pseudoistruzioni" o "direttive assembler", utilissime per controllare diverse operazioni connesse con il processo di assemblazione. Per la cronaca, la sola direttiva riconosciuta dall'asemblatore del BASIC è la "EQU" (nelle varianti EQUB, EQUW, EQUD, EQUS), utile per assegnare un valore (byte, word, double word o stringa) ad un simbolo definito dall'utente.

ADE+ riconosce invece circa una cinquantina di pseudo operazioni, che si possono suddividere in direttive di assemblazione, di definizione dati, di assemblazione condizionale, di controllo file.

Le macro

Tra le direttive di assemblazione, di particolare interesse è la possibilità di gestire "Macro Istruzioni" (di qui la denomi-

nazione di Macro Assemblatore). Definire una Macro significa associare ad una nome inventato dall'utente (l'identificatore della Macro), un intero gruppo di istruzioni, cosicchè usare questo nome all'interno del programma assembly, equivale al riscrivere il gruppo d'istruzioni in questione. Vi è inoltre la possibilità di passare dei "parametri" alla Macro, in questo modo, facendo largo uso di Macro Istruzioni, si scrivono programmi più leggibili, quasi come se si usasse un linguaggio ad alto livello.

Un breve esempio

Vediamolo attraverso un esempio: supponendo che in un programma sia necessario richiamare più volte la funzione VDU, cosa che in L.M. si svolge con due operazioni: caricando A con il carattere da stampare, e chiamando la routine OSWRCH; risulta comodo definire una Macro, chiamandola per es. VDU:

VDU MACRO ; Definizione della Macro "VDU"

LDA #@1 ; carica in A il primo parametro

JSR OSWRCH ; chiama la routine di stampa

ENDM ; fine della definizione di Macro

Lo strano simbolo @1 sta per "il primo parametro della Macro" (@2 per il secondo, @3 per il terzo...); a questo punto, definita la Macro, se vogliamo stampare qualcosa nel nostro programma, non dobbiamo far altro che richiamarla: per scrivere una "W" basta fare: VDU "W"; per cambiare modo schermo : VDU 22 VDU modo Ora vogliamo di più: se il nostro programma deve cambiare spesso modo grafico, conviene definire una Macro per il modo:

MODE MACRO
VDU 22
VDU @1
ENDM

abbiamo cioè una Macro che richiama un'altra Macro! Ora per cambiare modo è sufficente scrivere ad es. MODE 3 e l'Assemblatore automaticamente si preoccuperà di sostituire la parola MODE con l'opportuna sequenza di codici definita nella Macro!

Qualcuno avrà notato che l'etichetta OSWRCH non è stata definita: ADE+ assegna all'inizio tutti i valori corretti ad identificatori come OSWRCH, OSBYTE, OSWORD, che pertanto sono tutti identificatori predefiniti; comodo vero? A questo punto nulla vieta di scrivere una raccolta di Macro di utile e frequente utilizzo: sarà possibile racchiuderle in un file che fungerà da libreria per il nostro Assemblatore. Ed a questo hanno già pensato i signori della System Software, infatti, sul disco di ADE+ oltre a dei programmi di esempio, esiste già pronta un' intera libreria di Macro nel file 'MACLIB'.

Il linker

Sempre fra le direttive di asssemblazione, ve ne sono alcune dedicate alla gestione di moduli di programma ed all'import/export di siboli esterni: questo discorso porta dunque a parlare del Linker.

Normalmente, quando si scrive un programma in linguaggio assembly, si specifica, tramite la direttiva ORG, l'indirizzo da dove questo programma dovrà partire in memoria; quindi si chiama l'Assemblatore, ed il programma è pronto per essere eseguito. Dovendo intraprendere però la stesura di qualcosa di più grande, appare logico suddividere questo mega programma in unità fondamentali, che chiameremo moduli; se tutto il programma non riesce a stare nella memoria disponibile per l'Assemblatore, sarà sempre possibile farlo stare un modulo alla volta. Tra i tanti moduli, ve ne potrà essere uno che funge da libreria, contenendo al suo interno tutta una serie di routine matematiche, grafiche, e così via; a questo faranno riferimento tutti gli altri moduli che necessitano di tali routine.

É possibile dunque scrivere ed assemblare i diversi moduli uno alla volta, senza però specificare un indirizzo di partenza per mezzo della ORG: l'Assemblatore produrrà un "codice rilocabile", che ovviamente non potrà essere direttamente eseguito, non essendo stato scritto per lavorare in nessun punto preciso della memoria. Assemblati dunque tutti i moduli del nostro programma, ci troviamo ad avere tanti codici rilocabili che così come sono non possono essere di alcuna utilità. Ora entra in ballo il Linker: si tratta di un programma che "ricuce" tra di loro tutti i moduli che gli vengono dati in pasto. Un codice rilocabile contiene al suo interno anche tutti i riferimenti necessari per il collegamento di quel modulo con l'esterno, ed è compito del Linker sfruttare queste informazioni per effettuare tutti i collegamenti tra i vari moduli e con il modulo di libreria, assegnare al programma un opportuno indirizzo di inizio e rilocare i diversi moduli nelle posizioni che competono a ciascuno.

Alla fine di tutto ciò il programma è pronto per girare! Questo lavorio potrà sembrare macchinoso, ma porta ad innumerevoli vantaggi quando ci si trova a trattare con programmi che, scritti in unica soluzione, sembrerebbero interminabili. Bisogna fare una modifica? Non serve riassemblare tutto il programma da capo, basta alterare il solo modulo interessato. Vediamo un piccolo esempio: MODULE Scribacchino text EXT ; "text" è un simbolo esterno SYSEXEC ENT ; ENT sta per "Entry point" LDX #-1 :ciclo INX LDA text,X ; prende un carattere alla volta BEQ :fine ; se è CHR 0 allora esce dal ciclo JSR OSASCI ; chiama la routine di stampa JMP :ciclo :fine RTS

Questo moduletto prende qualunque stringa si trovi all'indirizzo 'text' e la mostra sullo schermo. Problema: text non è definito, o meglio, è definito come 'EXT' cioè come riferimento ad un modulo esterno. Da notare l'entrata del programma: SYSEXEC è un nome speciale che viene riconosciuto dal Linker come punto d'ingresso principale per il programma. Ci serve un altro modulo dove mettere il testo da scrivere: MODULE Gallina ; contiene il testo per il modulo Scribacchino text ENT STR "Meglio l'ovetto " STR "oggi che la " STR

"gallina domani! " BRK ; l'istruzione BRK produce un codice ASCII 0 Ogni modulo deve essere assemblato separatamente: bisogna dare ad ADE+ i seguenti comandi: ASM parte1=Scribacchino ASM parte2=Gallina Ed ora linkare: LINK progr=parte1,parte2

Il Linker fa sentire la sua presenza informando del successo avuto dalla operazione: sul disco ora è presente un file di nome "progr" contenente il codice oggetto eseguibile. Adesso il programma è finalmente pronto: lo si può lanciare con il comando *RUN progr ed ecco subito apparire sullo schermo una nota massima...

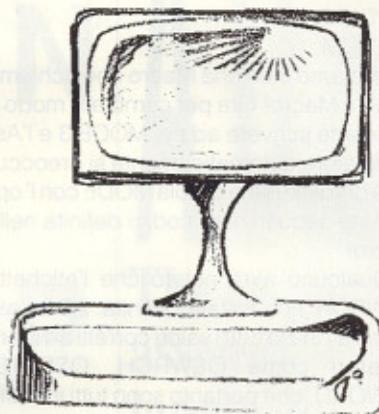
La MMU

Finora non se ne è parlato, ma tutte queste operazioni si sono svolte sotto lo sguardo vigile della MMU: la Memory Management Unit. Si tratta di un programma supervisore, che si occupa di sfruttare al meglio tutta la memoria a disposizione del PC128S, gestendo delle zone tampone dette appunto "buffer", ed impedendo che operazioni sconsiderate vadano ad alterare i contenuti delle zone riservate. Il primo contatto dell' utente con ADE+ si ha proprio con il livello comandi della MMU: da qui è possibile richiamare l'editor con il comando EDIT, l'Assemblatore con ASM, il Linker con LINK, il Debugger con DEBUG.

Il Debugger

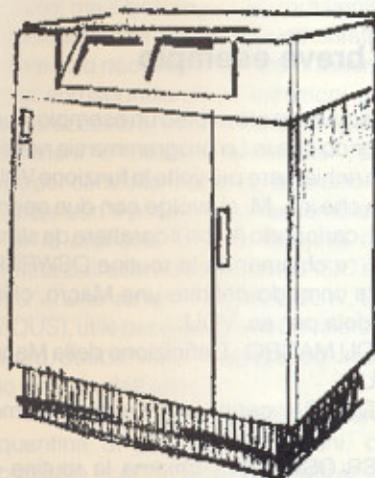
Parliamo dunque di quest'ultimo: Debugger è uno strumento praticamente indispensabile per curiosare nella memoria di un computer, visualizzando blocchi di byte, ed alterando contenuti di locazioni. Quello che appare all'utente è una specie di pannello di comando colorato, dove in alto sono visualizzati tutti i registri del processore con il loro contenuto, il Program Counter con l'istruzione corrente, lo stato dei vari flag. Al centro vi è una sorta di finestra aperta sulla memoria del PC128S: vi sono visualizzati 64 byte, a partire dall'indirizzo contenuto in un puntatore che può venir spostato a piacimento. In basso vi è una riga libera per l'introduzione dei comandi.

Si tratta per lo più di comandi che si attivano con la pressione di un solo tasto: è possibile andare "su e giù" con il puntatore di memoria, esplorando diverse zone di RAM o ROM, è possibile scegliere il for-

mato di visualizzazione, infatti i contenuti delle locazioni di memoria possono essere mostrati sotto forma di numeri esadecimali o attraverso i codici ASCII corrispondenti (molto utile per la ricerca di frasi e messaggi).

Di fondamentale importanza è la possibilità di "disassemblare" il contenuto della memoria: i valori numerici vengono ritrasformati in codici mnemonici, ottenendo una specie di programma in linguaggio assembly (senza però le etichette ed i commenti!); come se non bastasse vi sono comandi per la modifica di locazioni o gruppi di byte: quanto basta per accontentare gli smanettoni. Debugger però è soprattutto uno strumento per la verifica del funzionamento di routine in linguaggio macchina, non manca dunque la possibilità del "single step", dell'eseguire cioè un programma una istruzione alla volta, stando a guardare quello che succede. Si possono inserire inoltre fino ad otto "break points", cioè punti di controllo all'interno di un codice macchina: si lancia il programma, e quando questo arriva ad un break point, esso si arresta temporaneamente, ritornando il controllo all'utente che può modificare il contenuto di variabili o registri, per poi eventualmente proseguire.

Quindi...

Debug è proprio quanto basta per chiudere in bellezza con ADE+, sebbene questo pacchetto risulti sempre aperto a nuove eventuali espansioni (disponendo di qualche coprocessore, si potrebbe collegare un cross assembler...). Ovviamente quanto visto fino ad ora è solamente un assaggio delle potenzialità di ADE+, assaggio che avrà lasciato scontento sia il lettore principiante, che quello super esperto-mega informato (che di certo starà sbadigliando), ma avrà speriamo lasciato l'acquolina in bocca a tutti coloro i quali, già in possesso dei punti di partenza, desiderino fare un passo in avanti nell' utilizzo del linguaggio macchina.

Per questi, ma anche per i più esperti, ADE+ si rivelerà uno strumento veramente completo ed in grado di soddisfare qualsiasi esigenza. L' intero pacchetto è fornito su disco, e va caricato in memoria, dove occupa tre banchi di "Sideways RAM"; sullo stesso disco sono inoltre presenti programmi di esempio ed una libreria di Macro. Il tutto ovviamente è corredato da un bel manuale di riferimento (in inglese), dove vengono descritte a fondo tutte le possibilità di ADE+; vi è inoltre la possibilità di richiedere una guida avanzata, dove sono spiegati i dettagli tecnici sulla struttura dell'Assemblatore e del Linker.

Un elogio dunque alla System Software Ltd. per aver prodotto uno strumento degno di reggere il paragone con i più evoluti assemblatori presenti per macchine ad otto bit, e buon lavoro a tutti quelli che si accingeranno ad adoperarlo!

FAX1

CON VOI .... SEMPRE E OVUNQUE

L'Olivetti Prodest ha recentemente immesso sul mercato il Fax 1; questa macchina si presenta, agli occhi dell'acquirente, di ridotte dimensioni e si rimane piacevolmente sorpresi appena viene provata.

FAX 1 sarà da oggi il vostro nuovo assistente che vi seguirà ovunque, potrete trasmettere e ricevere personalmente ed in modo molto semplice i documenti. Data la sua compattezza le dimensioni risultano estremamente ridotte(36 x 18 x 8 cm.) con un peso complessivo di 3,3 Kg. e può essere comodamente contenuta anche in una valigetta 24 ore.

FAX 1 è allineato allo standard di comunicazione più diffuso nel mondo: il Gruppo III, che rappresenta il 95% del mercato mondiale fac-simile, questa sua peculiarità è una garanzia per comunicare sempre e con tutto il mondo. Di facile installazione è anche semplicissima da usarsi. La macchina non richiede alcuna manutenzione particolare ed è in grado di trasmettere e ricevere uno o più documenti con formato A4 in solamente 40 secondi premendo semplicemente un tasto. La stampa, di ottima qualità, è di tipo termico.

FAX 1 è un fac-simile di elevata qualità e tecnologicamente avanzatissimo viene venduto comunque ad un prezzo molto contenuto. La macchina può essere utilizzata anche come una comune copiatrice, dal momento che è in grado di effettuare istantaneamente copie di documenti con estrema rapidità; per utilizzarla in questa modalità si dovrà, come prima operazione, scollegare l'accoppiatore acustico.

Per operare vi sarà sufficiente un telefono ed una presa di corrente e la trasmissione sarà istantanea. Dovunque voi siate, in riunione, in albergo, potrete trasmettere e ricevere personalmente qualsiasi documento, con praticità e riservatezza.

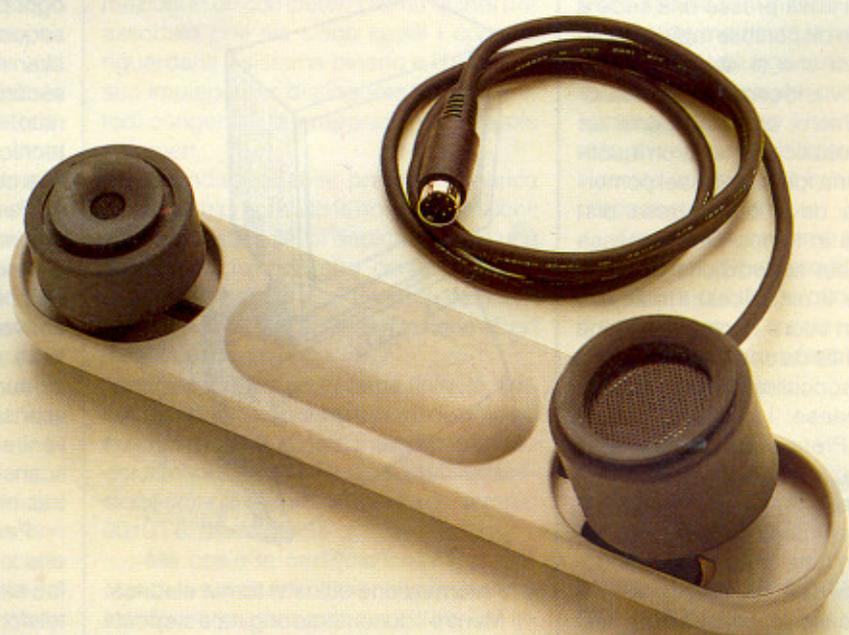
Dovrete far attenzione a non posizionare il FAX 1 vicino ad altoparlanti, televisori, radio, ecc. o di esporlo ai raggi diretti del sole; gli ambienti in cui operate non dovranno essere nè umidi nè polverosi, in breve dovrete usare le precauzioni che utilizzate normalmente nell'uso del computer. Per la pulizia della macchina basterà utilizzare un panno morbido.

Per quanto riguarda la connessione con il telefono è previsto oltre al normale collegamento diretto con la linea telefonica, l'utilizzo di un accoppiatore acustico che ne consente l'uso dovunque sia disponibile un telefono ed una presa di corrente. La selezione è possibile sia in modo manuale che in quello automatico. Se viene utilizzato l'accoppiatore acustico la velocità del modem è di 2400 (baud rate) mentre con l'accoppiamento diretto è possibile scegliere tra 4800 e 2400 (baud rate). Il FAX 1 è accompagnato da un manuale operativo di immediata comprensione, nel quale ogni informazione è collegata ad un disegno esplicativo.

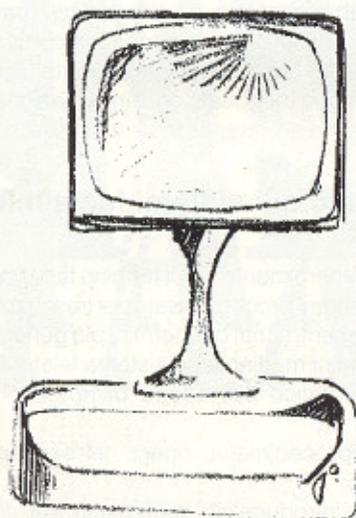
Cenno sui sistemi fac-simile

Generalmente con il termine fac-simile si intende il processo usato per trasmettere documenti scritti o, in termini più generali, immagini mediante un sistema telefonico o telegrafico allo scopo di riprodurli a distanza.

Il procedimento opera nel seguente modo:

si introduce il documento da trasmettere all'ingresso dell'unità di trasmissione del sistema fac-simile che opera la conversione dell'informazione visiva in un segnale elettrico corrispondente. Questi segnali elettrici modulano un trasmettitore, in questo modo

Accoppiatore acustico

l'informazione viene inviata al ricevitore, il quale converte l'informazione ricevuta nello stesso tipo di segnale elettrico generato presso il trasmettitore, e trasforma questa configurazione del segnale elet-

trico in una copia del documento originale.

Questo sistema è particolarmente adatto per la trasmissione di documenti di varia origine.

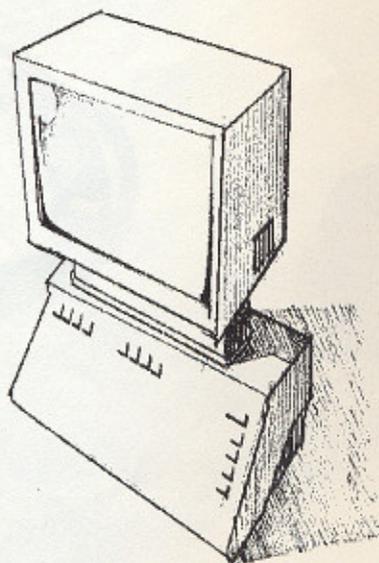
L'uso più ovvio di questo sistema si ha nel mondo degli affari, quando un qualsiasi documento si trova presso una sede e deve essere reso disponibile rapidamente in un'altra senza che si abbia il tempo sufficiente per inviarlo con i mezzi tradizionali (posta, corriere, ecc.). Si pensi ad esempio alla pubblicazione di un quotidiano quando una foto ripresa nel pomeriggio a Londra deve essere resa disponibile in Italia in tempo perchè possa essere pubblicata sull'edizione del mattino del giornale. In alcuni casi, il materiale fotografico di un intero quotidiano, viene inviato in fac-simile da un centro editoriale centrale alle succursali nelle diverse regioni di un paese. I servizi di notizie dell'Associated Press e della United Press International utilizzano già da molto tempo questi sistemi per trasmettere notizie e foto sui vari avvenimenti ai giornali abbonati al servizio in tutte le parti del mondo.

Un'altra applicazione importante e comune del fac-simile consiste nel trasmettere mappe della situazione metereologica alle stazioni TV ed agli aereoporti. È così possibile raccogliere dati sul tempo provenienti dai punti di rilevamento, elaborarli e distribuire a tutti gli interessati nelle varie parti del mondo le mappe metereologiche così ottenute.

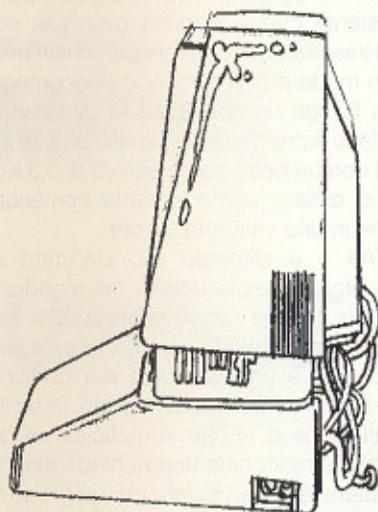
In vari paesi le autorità giudiziarie utilizzano il fac-simile per la trasmissione di impronte digitali, fedine penali, foto segnaletiche, patenti d'automobile, dati di immatricolazione di autoveicoli, verbali di polizia, carte di identità, ecc. in modo da poter controllare rapidamente eventuali individui sospetti.

Un altro impiego per il fac-simile è la distribuzione di documenti ed archivi di biblioteche da una sede centrale a succursali sparse per il mondo. In questo modo, una biblioteca centrale può rendere disponibili, al pubblico di lontane località, copie di libri originali rari e costosi o di documenti che non possono essere spostati.

Infine, il fac-simile può essere impiegato in applicazioni legali per inviare copie di documenti, (certificati di nascita, assegni bancari, testamenti, ecc.), che devono essere resi disponibili in una località distante dal posto di origine nel più breve tempo possibile.

Dopo aver visto quali siano gli elementi base di un sistema fac-simile, analizziamo ora come avviene la codifica

dell'informazione ottica in forma elettrica.

Mentre il documento originale si sposta entro l'unità di trasmissione, un fascio luminoso con intensità variabile in relazione ai chiari e scuri dell'immagine o del documento esaminato, viene trasformato in un segnale elettrico. Per effettuare questa conversione si ricorre ad appositi dispositivi (es. fotodiodo, fotomoltiplicatore, ecc.). Poi il fotorilevatore viene foca-

lizzato in modo che possa esaminare solo una porzione ristretta del documento, naturalmente tutte le parti del documento o immagine, devono essere successivamente prese in esame.

Il modo più comune per ottenere che ogni punto della carta venga esaminato sequenzialmente, è quello di effettuare la scansione dei punti lungo la superficie secondo uno schema regolare ed ordinato. Non ci soffermeremo sulle varie tecniche di scansione, basterà sapere che ce ne sono di vario tipo.

Per riprodurre la copia del documento originale, la configurazione della tensione elettrica deve essere riconvertita in forma visibile. Le tecniche disponibili per questa conversione differiscono fra loro per praticità e costo. Tutte però richiedono in comune la presenza di un meccanismo di scansione presso il ricevitore, che possa venire sincronizzato con il dispositivo di scansione esistente presso il terminale di trasmissione.

Per concludere, va sottolineato il fatto che la trasmissione utilizzata dai sistemi fac-simile avviene generalmente su linee telefoniche sebbene in alcuni casi sia necessario ricorrere alla trasmissione via radio, ad esempio, per l'invio di mappe metereologiche a navi in navigazione.